BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Sebagai landasan akademis, penelitian ini merujuk pada beberapa studi relevan yang telah dilakukan sebelumnya. Kajian pertama dilakukan oleh Rudi Pratama dan tim (2021) mengenai pengembangan aplikasi *Web* untuk manajemen peminjaman ruangan. Hasil penelitian menunjukkan bahwa solusi digital dapat meningkatkan efektivitas pengelolaan ruangan melalui fitur notifikasi otomatis dan antarmuka yang *user-friendly* [6].

Studi lain oleh Daffa Ramadan dkk. (2022) berhasil mengimplementasikan sistem terpadu untuk *reservasi* ruangan di Institut Bisnis dan Informatika Kosgoro, dengan capaian peningkatan efisiensi penggunaan sumber daya hingga 40% [7]. Temuan serupa juga diperoleh Rizky Ridho Prasetyo dkk. (2018) yang menggunakan pendekatan SDLC dan pengujian *black-box*, menunjukkan peningkatan 30% efisiensi manajemen ruang di UPN Veteran Jakarta [8].

Penelitian oleh Rakhman dkk. (2021) tentang Sistem Informasi Kemahasiswaan (SIKEMAS) di Politeknik Harapan Bersama memberikan bukti empiris bahwa penerapan *framework CodeIgniter* dengan *database MySQL* mampu menangani kompleksitas data akademik untuk lebih dari 1000 mahasiswa [9]. Demikian pula studi Afriliana dkk. (2017) yang mengembangkan sistem informasi berbasis *web* untuk Sasana Anak Suko

Mulyo, berhasil menggantikan sistem manual berbasis papan pengumuman dengan solusi digital yang lebih efisien [10].

Inovasi sistem peminjaman terintegrasi juga ditunjukkan oleh Ilham Khasbi & Fajar Nugraha (2016) melalui fitur notifikasi SMS yang mampu mengurangi keterlambatan pengembalian barang hingga 25% [11]. Temuan ini diperkuat oleh evaluasi Tim Pustaka Karya (2022) yang mengungkap tingkat kepuasan pengguna sistem digital mencapai 85%, jauh lebih tinggi dibanding metode konvensional [12].

Berbagai temuan tersebut tidak hanya memberikan kerangka konseptual yang kuat, tetapi juga bukti empiris tentang efektivitas solusi berbasis *web*. Dalam penelitian ini, metode dan teknologi yang telah teruji pada studi-studi sebelumnya akan diadaptasi dengan penyesuaian spesifik untuk konteks peminjaman ruangan di Politeknik Harapan Bersama, dengan tetap mempertimbangkan karakteristik unik institusi dan kebutuhan pengguna.

2.2 Landasan Teori

Berikut penyempurnaan teks dengan gaya penulisan yang lebih natural dan variatif:

2.2.1 Visual Studio Code

Sebagai salah satu *tools* pengembangan perangkat lunak, *Visual Studio Code* (VS Code) menawarkan solusi komprehensif bagi *developer*. Dikembangkan oleh Microsoft sebagai produk *open-source*, *software* ini kompatibel dengan berbagai *platform* termasuk *Windows*,

macOS, dan *Linux*. Meski memiliki ukuran yang relatif ringan, *VS Code* menyediakan beragam fitur canggih seperti *syntax highlighting, intelligent code completion, debugging tools*, dan *integrasi* dengan *Git* [13]. Dukungannya terhadap berbagai bahasa pemrograman - mulai dari *JavaScript*, *Python*, hingga C++ - membuatnya menjadi pilihan populer di kalangan pengembang *web* [14]. Keunggulan utamanya terletak pada kemampuan untuk mengenali struktur kode dan menyesuaikan tampilan secara otomatis, meningkatkan produktivitas dalam penulisan kode.

2.2.2 Database

Dalam konteks pengembangan sistem informasi, basis data memegang peranan krusial sebagai tempat penyimpanan data terstruktur. Konsep ini memungkinkan pengelolaan informasi secara sistematis melalui operasi *Create, Read, Update, Delete* (CRUD) [15]. Implementasi basis data yang efektif dapat menyederhanakan proses pengambilan keputusan dengan menyajikan data yang telah diolah menjadi informasi bernilai. Secara fundamental, basis data merupakan kumpulan tabel yang saling berelasi, dirancang untuk mengoptimalkan penyimpanan dan pengelolaan informasi.

2.2.3 Framework CodeIgniter

CodeIgniter menawarkan solusi pengembangan *web* yang efisien melalui pendekatan *Model-View-Controller* (MVC). Sebagai *framework PHP open-source*, *CodeIgniter* menyediakan berbagai

library dan *helper* yang mempercepat proses *development* [16]. Keunggulan utamanya terletak pada kemudahan implementasi dan dokumentasi yang lengkap, memungkinkan *developer* fokus pada logika bisnis daripada membangun komponen dasar dari nol. Fitur-fitur seperti *query builder* dan *form validation* membantu mengurangi redundansi kode sekaligus meningkatkan keamanan aplikasi.

2.2.4 Arsitektur Sistem Informasi

Sistem informasi merupakan integrasi antara komponen teknologi, prosedur, dan sumber daya manusia untuk mengolah data menjadi informasi bernilai [17]. Dalam praktiknya, sistem ini berfungsi sebagai pendukung pengambilan keputusan melalui transformasi data mentah menjadi *insight* yang *actionable*. Implementasi yang tepat dapat meningkatkan efisiensi operasional dan memberikan *competitive advantage* bagi organisasi.

2.2.5 Paket Pengembangan XAMPP

XAMPP menyederhanakan proses setup lingkungan pengembangan *web* dengan menggabungkan *Apache, MySQL, PHP, dan Perl* dalam satu paket terintegrasi [18]. Solusi *all-in-one* ini menghilangkan kebutuhan konfigurasi manual masing-masing komponen, memungkinkan *developer* langsung fokus pada pembuatan aplikasi. Kemampuannya untuk berjalan di berbagai sistem operasi menjadikannya pilihan ideal untuk pengembangan *cross-platform.*

2.2.6 Pemodelan Sistem dengan UML

Unified Modeling Language (UML) telah diakui sebagai bahasa pemodelan standar dalam rekayasa perangkat lunak yang memungkinkan pengembang untuk membuat representasi visual dari desain sistem. Seperti halnya cetak biru yang menjadi panduan penting dalam konstruksi bangunan, diagram *UML* berperan sebagai fondasi konseptual sebelum pembangunan sistem perangkat lunak dimulai [19]. Bahasa pemodelan ini menyediakan beragam jenis diagram yang mampu menggambarkan aspek struktural, perilaku, dan dinamika suatu sistem. Dalam praktiknya, terdapat empat diagram utama yang paling sering dimanfaatkan: *Diagram Use Case* untuk memetakan interaksi antara pengguna dengan fungsionalitas sistem, Diagram Aktivitas yang merepresentasikan alur proses bisnis secara rinci, Diagram Sekuens untuk menelusuri pertukaran pesan antar objek dalam suatu skenario tertentu, serta Diagram Kelas yang mendefinisikan arsitektur statis sistem melalui kelas, atribut, dan relasinya [19]. Pemilihan diagram *UML* yang tepat sangat bergantung pada kompleksitas sistem dan tahapan pengembangan, dimana diagram berbasis perilaku seperti use case lebih dominan digunakan pada fase analisis kebutuhan, sementara diagram struktural seperti diagram kelas lebih banyak diaplikasikan pada tahap desain teknis.

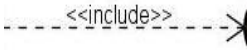
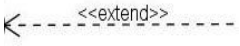
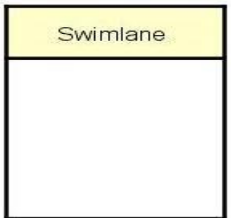
Terdapat beberapa diagram *UML* yang sering digunakan dalam pengembangan sebuah sistem yaitu:

1. Diagram *Use Case*

Diagram ini merepresentasikan kebutuhan fungsional sistem dengan menampilkan aktor (baik pengguna manusia maupun sistem *eksternal*) dan *use case* (fungsi yang disediakan sistem). Setiap *use case* menunjukkan bagaimana aktor berinteraksi dengan sistem untuk mencapai tujuan tertentu. Simbol-simbol yang digunakan dalam diagram *use case* dijelaskan pada Tabel 2.1 berikut:

Tabel 2. 1 *Use case Diagram*

No	Simbol	Deskripsi
1	*Use case* 	*Use case*: Abstraksi dari interaksi antara sistem dan aktor
2	Aktor / *actor* 	Aktor: Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan sistem
3	Asosiasi / *association*	*Association*: Abstraksi dari penghubung antara aktor dengan *use case*
4	*Generalization* 	*Generalization*: Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan *use case*

No	Simbol	Deskripsi
5	*Include* 	Menunjukkan bahwa suatu *use case* selalu merupakan fungsionalitas dari *use case* lainnya
6	*Extend* 	Menunjukkan bahwa suatu *use case* merupakan tambahan fungsional dari *use case* lainnya jika suatu kondisi terpenuhi
7	*Swimlane* 	*Swimlane*, pembagian *activity diagram* untuk menunjukan siapa melakukan apa
8	*Colaboration* 	Interaksi aturan-aturan dan elemen-elemen lain yang bekerja sama untuk menghasilkan perilaku yang lebih besar daripada sekadar penjumlahan dari masing-masing elemennya (sinergi*).*

2. Diagram Aktivitas

Mengilustrasikan alur kerja sistem melalui serangkaian aktivitas dan keputusan. Diagram ini sangat berguna untuk memodelkan proses bisnis yang kompleks dengan menunjukkan aliran kontrol dari satu aktivitas ke aktivitas lainnya. Simbol-simbol yang digunakan dalam *diagram activity* dijelaskan pada Tabel 2.2 berikut:

Tabel 2. 2 Diagram Activity

NO	GAMBAR	NAMA	KETERANGAN
1		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka satu ling berinteraksi satu sama.
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
3		*Initial Node*	Bagaimana objek dibentuk dan diakhiri.
4		*Activity Final Node*	Bagaimana objek dibentuk dan diakhiri.
5		*Decision*	Digunakan untuk suatu keputusan/tindakan yang harus diambil pada kondisi tertentu.

NO	GAMBAR	NAMA	KETERANGAN
6		*Line Connector*	Digunakan untuk menghubungkan satu simbol dengan simbol lainnya.

3. Diagram Sekuens

Menjelaskan bagaimana objek-objek dalam sistem saling berinteraksi melalui pertukaran pesan dalam suatu skenario tertentu. Diagram ini sangat membantu untuk memahami dinamika sistem berdasarkan urutan kejadian. Simbol-simbol yang digunakan dalam *diagram sekuens* dijelaskan pada Tabel 2.3 berikut:

Tabel 2. 3 *Diagram Sekuens*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menggambarkan orang atau sistem eksternal yang sedang berinteraksi dengan sistem yang dikembangkan.
2		*Entity Class*	Menggambarkan entitas yang menyimpan data atau objek yang berkaitan dengan data yang akan dilakukan.

NO	GAMBAR	NAMA	KETERANGAN
3		*Boundary Class*	Menggambarkan batasan sistem (antarmuka pengguna), seperti tampilan form atau halaman.
5		*A Focus of Control & A Life Line*	Menggambarkan waktu hidup (lifeline) dari sebuah objek, dari awal hingga akhir interaksi.
6		*A Message*	Menggambarkan pesan atau interaksi antar objek, seperti pemanggilan metode atau komunikasi data.

4. Diagram Kelas

Merupakan inti dari desain berorientasi objek yang menampilkan kelas-kelas beserta atribut, operasi, dan hubungan antar kelas seperti pewarisan, asosiasi, dan ketergantungan. Simbol-

simbol yang digunakan dalam diagram kelas dijelaskan pada Tabel 2.4 berikut:

Tabel 2. 4 Diagram Kelas

NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan di mana objek anak (descendent) mewarisi perilaku dan struktur data dari objek induk (ancestor).
2		*N-ary Association*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
3		*Class*	Representasi dari sebuah kelas, yaitu deskripsi dari urutan aksi-aksi dan atribut-atribut yang mendefinisikan objek.
4		*Collaboration*	Deskripsi dari interaksi objek-objek dalam sistem untuk menghasilkan suatu hasil yang terukur bagi actor.

NO	GAMBAR	NAMA	KETERANGAN
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek untuk merealisasikan tugasnya.
6		*Dependency*	Hubungan di mana perubahan yang terjadi pada suatu elemen mandiri (independent) akan memengaruhi elemen lain yang bergantung padanya (dependent).
7		*Association*	Hubungan yang menghubungkan antara satu objek dengan objek lainnya.

2.2.7 Black Box Testing

Black box testing atau *behavioral testing* adalah metode evaluasi perangkat lunak yang mengabaikan aspek implementasi internal sistem. Pendekatan ini menitikberatkan pada pengujian fungsionalitas berdasarkan spesifikasi kebutuhan, dimana penguji hanya

memperhatikan hubungan antara *input* yang diberikan dan *output* yang dihasilkan. Teknik ini biasanya melibatkan pengujian fungsional seperti *equivalence partitioning*, *boundary value analysis*, dan uji coba berbasis skenario *use case*. *Black box testing* umumnya dilakukan pada tahap akhir pengembangan untuk memvalidasi bahwa sistem berperilaku sesuai dengan harapan pengguna akhir dan memenuhi semua *requirement* yang telah ditetapkan sebelumnya.